Framework 7.2

Load Distribution Server

User's Guide

About Genesys

Genesys Telecommunications Laboratories, Inc., a subsidiary of Alcatel-Lucent, is 100% focused on software for call centers. Genesys recognizes that better interactions drive better business and build company reputations. Customer service solutions from Genesys deliver on this promise for Global 2000 enterprises, government organizations, and telecommunications service providers across 80 countries, directing more than 100 million customer interactions every day. Sophisticated routing and reporting across voice, e-mail, and Web channels ensure that customers are quickly connected to the best available resource—the first time. Genesys offers solutions for customer service, help desks, order desks, collections, outbound telesales and service, and workforce management. Visit www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies. The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the following regional numbers:

Region	Telephone	E-Mail
North and Latin America	+888-369-5555 or +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-1276-45-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-6361-8950	support@genesyslab.co.jp

Prior to contacting technical support, please refer to the *Genesys Technical Support Guide* for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the ***Genesys 7 Licensing Guide*.**

Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 72fr_us_lds_05-2008_v7.2.001.02

Table of Contents

Preface

Welcome to the *Framework 7.2 Load Distribution Server User's Guide*. This document introduces you to the concepts, terminology, and procedures relevant to Load Distribution Server (LDS).

This document is valid only for the 7.2 release of this product.

Note: For releases of this document created for other releases of this product, please visit the Genesys Technical Support website or request the Documentation Library CD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

This chapter has these sections:

In brief, you will find the following information in this guide:

- A high-level description of Load Distribution Server (LDS) and its uses
- A description of the distribution modes you can configure for LDS
- Procedures for installing and configuring LDS
- A description of all LDS configuration options
- Additional information about using LDS with Genesys Routing solutions
- Configuration guidelines for using LDS with Call Concentrator (CCon), which is part of the Genesys Reporting solution

Intended Audience

This guide is primarily intended for system administrators in contact centers. It assumes that you have a basic understanding of and familiarity with:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications.

- Network design and operation.
- Your own network configurations.

You should also be familiar with Genesys Framework architecture and functions.

Chapter Summaries

In addition to this opening chapter, this guide contains these chapters

- Chapter 1, "Overview," on page 13, contains a high-level description of what LDS is, the different distribution modes it can use, and how to configure LDS for these modes.
- Chapter 2, "Weighted Round Robin (WRR) Mode," on page 23 describes `Weighted Round Robin` (WRR) mode for LDS.
- Chapter 3, "Installation," on page 25, describes how to install LDS on Windows and UNIX/Linux platforms.
- Chapter 4, "Starting and Stopping LDS," on page 27, describes how to start and stop LDS.
- Chapter 5, "Configuration Options," on page 29, describes all of the configuration options available with LDS, as well as the log options specific to LDS.
- Chapter 7, "High-Availability (HA) Configuration," on page 57, describes high availability (HA) for LDS and associated T-Servers and clients.
- Chapter 8, "LDS Support (Load Distribution Mode) of Routing," on page 61, provides additional information about using Universal Routing Server as a client of LDS.
- Chapter 9, "LDS Support (Load Distribution Mode) of Call Concentrator," on page 71, provides recommendations for using Call Concentrator (CCon) and LDS.

New for LDS in Release 7.2

This section gives summary details of new features in release 7.2.

- **Support for new operating systems:**
  - Red Hat Linux version 4.0
  - HP Compaq TRU64/Alfa version 5.1B
  - Solaris/Sparc 10 32/64-bit version 2.10
  - Microsoft Windows Server version Windows 2003 64-bit. Support for Windows NT is discontinued.
- **Support for rolling upgrades:** In release 7.2, when a backup LDS connects, the primary LDS can mark all existing transactions as "not synchronized" and prevent the switchover request from Solution Control Interface. As soon as all marked transactions are removed, switchover becomes possible. This feature is controlled by configuration options `ha-dly-switchover` and `max-update-rate`—see page 36.
- **Enhanced extended HA client support:** In release 7.2 a new attribute `EventSequenceId` has been implemented to improve extended HA client support.
- **Enhanced handling of call extensions in events:** In release 7.2.002+, you can configure filtering of ConnID-related extensions in events `EventRegistered` and `EventAddressInfo`. This feature provides a more accurate report of ConnIDs to Receivers. Only ConnIDs of transactions already assigned to Receivers are now visible to Receivers. See "keep-ext-key" on page 37.
- **Configuration option `use-end-of-call`:** Configuration option `use-end-of-call` has been removed in the 7.2.002+ versions of LDS. See page 33.

Document Conventions

This document uses some stylistic and typographical conventions with which you might want to familiarize yourself.

Document Version Number

A document version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

71fr_us_05-2005_v7.2.001.00

You will need this number when you are talking with Genesys Technical Support about this product.

Type Styles

Italic

In this document, italic is used for emphasis, for documents' titles, for definitions of (or first references to) unfamiliar terms, and for mathematical variables.

Examples:

- Please consult the *Genesys 7 Migration Guide* for more information.
- *A customary and usual practice* is one that is widely accepted and used within a particular industry or profession.
- Do *not* use this value for this option.
- The formula, $x + 1 = 7$ where x stands for . . .

Monospace Font

A monospace font, which looks like `teletype or typewriter text`, is used for all programming identifiers and GUI elements.

This convention includes the *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages; the values of options; logical arguments and command syntax; and code samples.

Examples:

- Select the `Show variables on screen` check box.
- Click the `Summation` button.
- In the `Properties` dialog box, enter the value for the host server in your environment.
- In the `Operand` text box, enter your formula.
- Click `OK` to exit the `Properties` dialog box.
- The following table presents the complete set of error messages T-Server® distributes in `EventError` events.
- If you select `true` for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls.

Monospace is also used for any text that users must manually enter during a configuration or installation procedure, or on a command line:

Example:

- Enter `exit` on the command line.

Screen Captures Used in This Document

Screen captures from the product GUI (graphical user interface), as used in this document, may sometimes contain a minor spelling, capitalization, or grammatical error. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from

installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Square Brackets

Square brackets indicate that a particular parameter or value is optional within a logical argument, a command, or some programming syntax. That is, the parameter's or value's presence is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information. Here is a sample:

```
smcp_server -host [/flags]
```

Angle Brackets

Angle brackets indicate a placeholder for a value that the user must specify. This might be a DN or port number specific to your enterprise. Here is a sample:

```
smcp_server -host <confighost>
```

Related Resources

Consult these additional resources as necessary:

- Documentation for the solution with which you are using LDS (Routing or Reporting).
- *Genesys Migration Guide,* which contains a documented migration strategy for each software release. Please refer to the applicable portion of the guide, or contact Genesys Technical Support for additional information.
- The *Genesys Master Glossary* document, which ships on the Genesys Documentation Library CD. Provides a fairly comprehensive list of Genesys and CTI terminology and acronyms.
- The *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library CD and which provides a comprehensive list of the Genesys and CTI terminology and acronyms used in this document.
- The Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at: http://genesyslab.com/support.

Information on supported hardware and third-party software is available on the Genesys Technical Support website in the following documents:

- *Genesys 7 Supported Operating Systems and Databases*

- *Genesys 7 Supported Media Interfaces*

Genesys product documentation is available on the:

- Genesys Technical Support website at http://genesyslab.com/support.
- Genesys Documentation Library CD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

Chapter

1 Overview

This chapter describes four of the five different LDS distribution modes. A fifth mode (`Weighted Round Robin` mode) is described in Chapter 2, "Weighted Round Robin (WRR) Mode," on page 23. This chapter contains the following topics:

Concepts

Overview

Purpose

LDS is designed to increase system performance in contact center environments with high call volumes. LDS enables load sharing in situations where the total traffic of a large installation exceeds the capacity of individual Receivers. Using LDS with multiple Receivers also increases redundancy in a configuration.

Terminology

LDS mediates between *Senders* and *Receivers*.

A Sender is a T-Server. Any premise or network T-Server can be a Sender. Support for Media Server (MS) T-Servers is not available in release 7.2.

A Receiver is a T-Server client.

Smart Distribution of Events

LDS divides the traffic into manageable portions and distributes it among Receivers by using *smart distribution* of T-Server events (*T-events*). Smart distribution means that LDS correctly identifies all T-events related to a given interaction and passes them on to the appropriate Receiver(s).

Start/Stop Overview

At startup, LDS connects to all T-Servers for which it is configured, without waiting for Receiver connections. If it cannot establish a connection at that time with a T-Server at startup, LDS repeats the connection attempt after a Receiver connects to it. LDS stays connected to the T-Servers as long as it is running, but it unregisters from all DNs after the last Receiver disconnects from it.

Receiver Types

From the T-Server's perspective, the type of Receiver is unimportant.

The Receiver type that an LDS instance supports is dynamically defined during runtime by the first Receiver to succeed in connecting to LDS. If you need to configure LDS for different Receiver types within the same system, you must use a separate instances of LDS for each Receiver type.

Note: The type of client application that can be a Receiver differs according to the LDS mode of operation.

Modes of Operation

You can use LDS in any of the following four distribution modes:

- `Load Distribution` mode
- `TProxy` mode
- `Broadcast` mode
- `Single T-Server LDS` mode

These modes are described in the remainder of this chapter. More detailed information on specific configurations and usages can be found in later chapters of this document.

Application Types for LDS

You can configure an LDS with application type:

- `LoadDistributionServer`
- `T-Server`

Single T-Server Configuration

Prior to release 7.0, LDS could operate only in a multi-T-Server configuration. In this configuration and in normal `Load Distribution` mode, LDS connects to multiple T-Servers to broker messages to multiple Receivers (see Chapter 8, "LDS Support (Load Distribution Mode) of Routing," on page 61 for an illustration of how this mode operates with the Universal Routing Server (URS) routing client).

From release 7.0 onward, LDS can also operate in a single–T-Server configuration using the functionalities described in this chapter.

You can use the proxy functions described in this chapter only in a single–T-Server environment where an LDS is configured with application type `T-Server`.

distribute-mode Configuration Option

Configuration option `distribute-mode` specifies the default distribution mode of LDS.

distribute-mode

Default Value: `auto`

Valid Values: `auto, load, proxy`

`auto`	With value `auto`, LDS selects its own mode depending on its actual configuration environment. With LDS configured in Configuration Manager with application type `T-Server` and one T-Server added on the LDS `Connection` tab, LDS operates in `proxy` mode. For all other possible configurations, LDS operates in `Load Distribution` mode.
`load`	With value `load`, LDS is forced into `LoadDistribution` mode.
`proxy`	With value `proxy`, LDS acts as proxy between T-Server and clients

Changes Take Effect: Immediately

Specifies the default distribution mode of LDS.

Note: Genesys recommends that you do not make changes without careful preparation, because the effect on client applications of changing between `Proxy` and `LoadDistribution` modes could cause unexpected effects.

Resulting Distribution Modes

As a result of the two application types, and the ability to use LDS in a single–T-Server environment, four distribution modes are available, as shown in Table 1 on page 16.

.

Table 1: LDS Distribution Modes

LDS with Application Type...	Distribution Mode		
	`distribute-mode = load`	`distribute-mode = auto`	`distribute-mode = proxy`
LoadDistribution Server	Load Distribution	Load Distribution	Broadcast
T-Server (multiple T-Servers configured in `Connections` tab)	Load Distribution	TProxy	TProxy
T-Server (single T-Server configured in `Connections` tab)	Single T-Server LDS	TProxy	TProxy

Warning! Please keep in mind that `TProxy` mode is not compatible with the normal `Load Distribution` mode. You cannot mix these modes without breaking the load distribution model.

Load Distribution Mode

Concept

In `LoadDistribution` mode, LDS distributes requests (balances the load) among the Receivers by using smart distribution of T-events. Smart distribution means that LDS correctly identifies all T-events related to a given interaction and passes them on to the appropriate Receiver(s). Multiple–T-Server configurations can include any combination of premise and network T-Servers.

Supported Receiver Types

In `Load Distribution` mode, LDS supports only the following Receivers:

- URS release 6.5 and later
- CCon, release 6.1.001.12 and later

Illustration

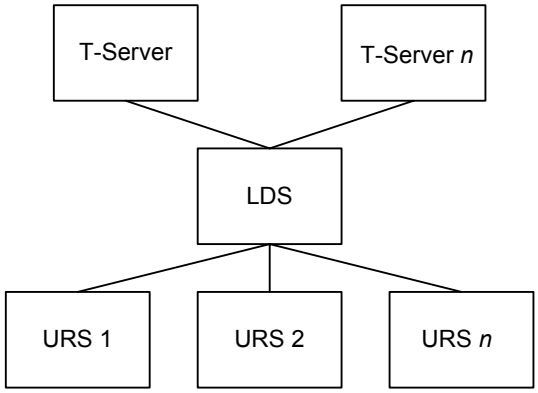

Figure 1: LDS in Load Distribution Mode

See Chapter 8, for samples of LDS configurations with Genesys Routing solutions.

Configuration

To configure LDS to operate in `Load Distribution` mode:

1. In Configuration Manager, create an `Application` object of type `LoadDistributionServer` using the `LDS_Server` template.
2. Install LDS, choosing `Load Distribution` mode during the installation procedure.
3. On the `Connections` tab of the new application, add a connection to each T-Server in the configuration.
4. On the `Switch` tab, add the same switch as is configured for each T-Server you added in Step 3.
5. On the `Options` tab, set the value of LDS configuration option `distribute-mode` to `load`.
6. Configure Receiver connections to point to the new LDS application.

TProxy Mode

Concept

Used in `TProxy` mode with a single T-Server, LDS reduces the amount of data transmitted over a WAN between remote T-Servers and T-Server clients in a central site. Instead of sending the same events multiple times—once for every client—T-Server sends the events a single time to a central LDS, which then distributes this event to all clients on the central site that are registered for a particular DN.

`TProxy` mode has the potential to reduce the volume of data carried over the WAN between T-Server and LDS to 1/*N* of the volume of data carried over the WAN when clients connect directly to T-Server. This can lead to a substantial cost reduction for the customer in environments where costs are based on number of bytes transmitted.

Supported Receiver Types

When LDS is in `TProxy` mode, Receivers can be of any type and combination of types, provided that they are T-Library compliant.

Illustration

Figure 2 illustrates how LDS in `TProxy` mode distributes TEvent 1. TEvent 1 is carried only once over the WAN before being distributed to all clients registered for the relevant DN. Redundant T-Server and LDS configurations are shown in red.

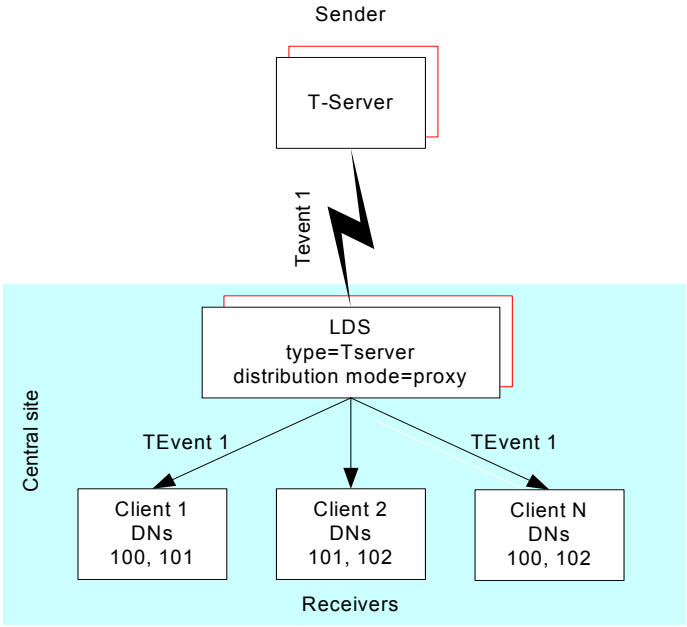

Figure 2: LDS in TProxy Mode

Configuration

To configure LDS to operate in `TProxy` mode:

1. In Configuration Manager, create an `Application` object of type `T-Server` using the `TProxy_Server` application template.
2. Install LDS, choosing `TProxy` mode during the installation procedure.
3. On the `Connections` tab of the new application, add a connection to a single T-Server.
4. On the `Switch` tab, add the same switch as is configured for the T-Server you added in Step 3.
5. On the `Options` tab, set the value of LDS configuration option `distribute-mode` to either `auto` or `proxy`.
6. Configure client connections to point to the new LDS application.

Note: From release 7.1, LDS configured in `TProxy` mode can control both active and passive Receivers sumiltaneously.

Tiered TProxy Mode

In this section the term *TProxy* is used to described an LDS configured in `TProxy` mode. So a client of such an LDS is a *TProxy client*, and so on.

Prior to release 7.1, the design of `TProxy` mode assumed either a TProxy directly connected to T-Server(s), or a TProxy-aware client that was able to access T-Server's switch configuration by using information from the LDS connection.

From release 7.1, you can create tiered configurations of TProxies, in which one TProxy can be a client of another TProxy. In this structure, a TProxy client can reach switch configuration details at the end of a chain of TProxies (or at any point within the chain where application type `T-Server` is found).

Such tiered configurations can be used to reduce the amount of network traffic flowing over the WAN between remote and central sites.

Example Configurations

Figure 3 on page 20 shows how tiered TProxies can be configured in different environments.

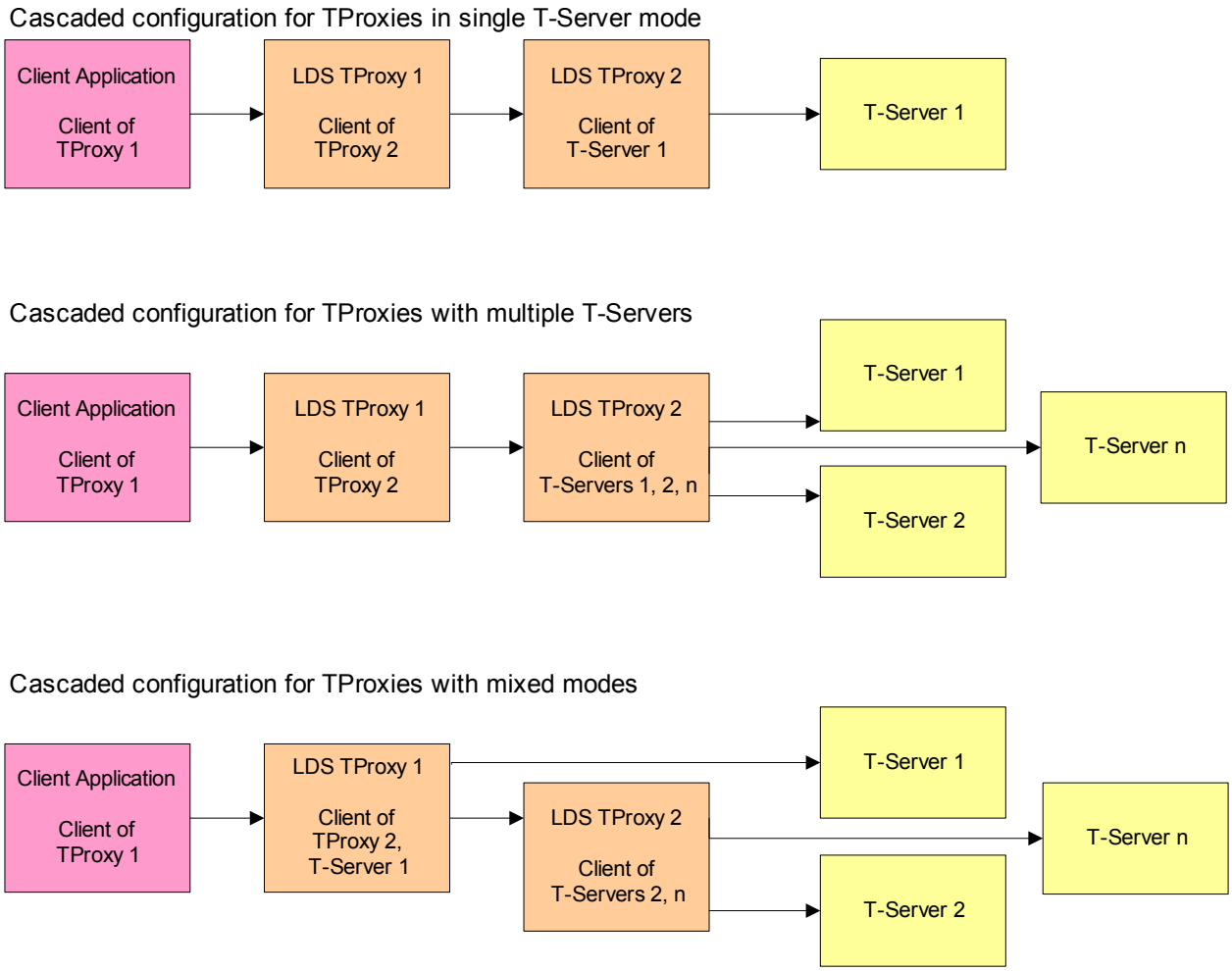

Figure 3: Tiered (Cascaded) Proxy Configuration Examples

Broadcast Mode

Concept

In `Broadcast` mode, LDS broadcasts events to any client that requests them. You can use any client capable of connecting to LDS in `Load Distribution` mode (that is, to LDS with application type `Load Distribution Server`) in `Broadcast` mode configurations. However, you cannot mix clients for `Broadcast` mode and clients for one of the load distribution modes (`Load Distribution` mode or `Single T-Server LDS` mode) on the same LDS.

LDS tracks all registrations and broadcasts, so it is not necessary for all clients to register for the same DNs.

Supported Receiver Types

In `Broadcast` mode, LDS only supports the following Receivers:

- URS release 6.5 and later

- CCon, release 6.1.001.12 and later

Illustration Figure 4 on page 21 illustrates how `Broadcast` mode operates. Redundant T-Server and LDS configurations are shown in red.

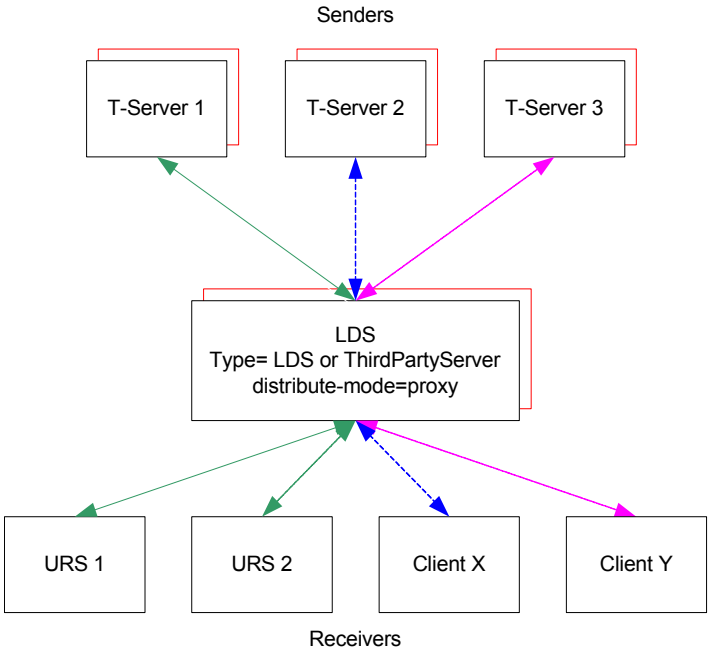

Figure 4: LDS in Broadcast Mode

Configuration To configure LDS to operate in `Broadcast` mode:

1. In Configuration Manager, create an `Application` object of type `LoadDistributionServer` using the `LDS_Server` application template.
2. Install LDS, choosing `Load Distribution` mode during the installation procedure.
3. On the `Connections` tab of the new application, configure the relevant connections to T-Servers.
4. On the `Options` tab of each instance of LDS, set the value of LDS configuration option `distribute-mode` to `proxy`.
5. Configure Receiver connections to point to the new LDS application.

Single T-Server LDS Mode

Concept When LDS is used in this mode, any Receiver can connect to LDS without modification, provided that LDS connects to only a single T-Server.

In earlier versions of LDS that do not support this mode of operation, you must modify Receivers to enable them to connect to multiple T-Servers—when connecting to LDS, the Receiver has to specify which T-Server it wants LDS

to connect to. When LDS connects to only one T-Server, you do not have to modify the Receiver.

Supported Receiver Types

With LDS in `Single T-Server LDS` mode, Receivers can be of any type, but must all be of the same type. If Receivers of different types are configured, load distribution becomes meaningless.

Illustration

Figure 5 illustrates how this mode operates.

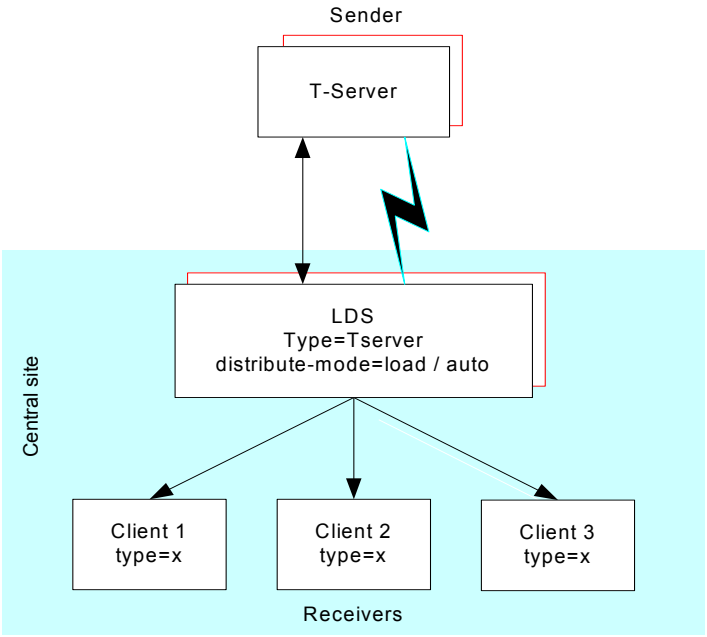

Figure 5: LDS in Single T-Server LDS Mode

Redundant T-Server and LDS configurations are shown in red.

Configuration

To configure LDS to operate in `Single T-Server LDS` mode, follow the steps below.

1. In Configuration Manager, create an `Application` object of type `T-Server` using the `TProxy_Server` application template.
2. Install LDS, choosing `TProxy` mode during the installation procedure.
3. On the `Connections` tab of the new application, add a connection to a single T-Server.
4. On the `Switch` tab, add the same switch as is configured for the T-Server you added in Step 3.
5. On the `Options` tab, set the value of LDS configuration option `distribute-mode` to `load`.
6. Configure Receiver connections to point to the new LDS application.

Chapter

2 Weighted Round Robin (WRR) Mode

This chapter provides information about the fifth of the five distribution modes, `Weighted Round Robin (WRR)` mode. See Chapter 1, "Overview," on page 13 for details of the other four.

This chapter includes the following sections:

Concept

`Weighted Round Robin (WRR)` mode is a variant of the standard `Load Distribution` mode that is achieved by configuring the Receiver application as described in this chapter. With `WRR` mode, you can alter the loading profile of an individual Receiver (or group of Receivers). So, for a given set of three Receivers, you can configure them to receive, for example, 50 per cent, 30 per cent and 20 per cent of the transactions, or any other proportion depending on your environment.

Receivers can also be grouped, and the loading profile can be set for a group.

Configuration Options

Set the two configuration options in this section in the `LDS` section of the relevant Receiver application. The section name within the Receiver application must be `LDS`.

Set the Receiver weighting using the configuration option `loading-coefficient`.

Use the `group-id` option to group a set of Receivers together. LDS treats such a group as a single Receiver.

loading-coefficient

Default Value: `100`
Valid Value: Any integer from `0-100`
Changes Take Effect: Immediately

Defines the relative loading coefficient for each specific Receiver for `Weighted Round Robin (WRR)` mode. For example, a set of three identical Receivers configured with loading coefficients `100`, `70`, and `30` receive 50 percent, 35 percent and 15 percent, respectively, of the total number of transactions.

If you use the default value (`100`) for all Receivers (`100`), you disable `WRR` and transactions are distributed in `Load Distribution` mode. If you set any nondefault value for any one Receiver (or Receiver group), you enable `WRR`.

Note: With value `0` (zero), a Receiver is excluded from transaction distribution. However, distribution to this Receiver is still possible, either when no other targets are available (for example, there are no more Receivers, or no more Receivers registered on a specific resource) or when this Receiver is available for noncontext distribution.

group-id

Default Value: `0`
Valid Value: Any integer from `0-65535`
Changes Take Effect: At LDS restart

Specifies whether this Receiver is part of a group. When you configure two or more Receivers with the same value for this option, LDS treats them as a single group and they receive the same event set.

Configure this option with caution. The fact that grouped Receivers receive identical call information has potential impacts on Receiver functionality; for example, duplicated records in Call Concentrator, calls being routed by two different URSs, and so on).

Note: You must ensure that both Receivers in the Configuration Manager primary/backup pair have the same value for the `group-id` option (if the option is configured). From release 7.1, the Configuration Manager primary/backup setting takes precedence.

Chapter

3 Installation

This chapter describes how to install LDS in UNIX/Linux and Microsoft Windows environments. It contains one section:

Installing LDS

This section describes the installation of LDS on UNIX/Linux and Windows.

Installing on UNIX/Linux

1. On the product CD, locate the appropriate shell script.
2. Run the script from the command prompt by typing `sh` followed by the file name.
3. When prompted, specify the `Host Name` of the computer on which to install LDS.
4. When prompted, specify the mode of LDS operation (`TProxy` or `LoadDistribution`).
5. When prompted, specify the following:
   - `Host Name` of the computer on which Configuration Server is running.
   - `Port` that client applications use to connect to Configuration Server.
   - `User Name` used to log in to the Configuration Layer.
   - `Password` used to log in to the Configuration Layer.
6. Depending on the mode selected in Step 4, installation displays a list of LDS applications of the relevant type configured for this host. Enter the number of the LDS application that you want to install.
7. Specify the full path of the destination directory into which you want to install LDS.
8. If asked, choose to install either the 32-bit or the 64-bit version, depending on your environment.

9. When prompted, specify the path to a valid license file.

As soon as the installation process is finished, a message appears announcing that the installation was successful. The process places LDS in the directory specified in Step 7..

Installing on Windows

1. From the product CD, locate and double-click the appropriate `Setup.exe` to start the installation.
2. When prompted, specify the mode of LDS operation (`TProxy` or `LoadDistribution`).
3. When prompted, specify the `Host` and `Port` of Configuration Server. Accept `ITCUtility` as the name of the Installation Configuration Utility application.
4. When prompted, specify the `User Name` and the `Password` used to log in to the Configuration Layer.
5. Confirm the `Host Name` of the computer on which to install LDS.
6. Depending on the mode selected in Step 2, installation displays a list of applications of the relevant type configured for this host. From the list, select the LDS application to install.
7. Specify the full path of the directory into which to install LDS.
8. Specify the `Program Folder` to which you want LDS added.
9. When prompted, specify the path to a valid license file.
10. Decide whether you want to install LDS as a Windows service. For more information, see the *Framework 7.2 Deployment Guide.*
11. When icons for LDS appear, click `Finish` to complete the installation

Note: LDS supports Configuration Server backup and Configuration Server proxy configurations.

Chapter

4 Starting and Stopping LDS

This chapter describes how to start and stop LDS. It contains the following topics:

Starting LDS

Ensure that DB Server and Configuration Server are running. If you are using the Management Layer, ensure that all of its components are running, including Solution Control Interface. For instructions on starting and stopping LDS via the Management Layer, see *Solution Control Interface Help* in the Genesys Framework 7.2 documentation.

Command-Line Parameters

You must specify command-line parameters (also called command-line arguments) to operate LDS, whether you are operating it manually or with the Management Layer. In manual operation, you either enter parameters directly on a command line or invoke them from a batch file, which is invoked in turn either directly on a command line or via a shortcut on the Windows `Start` menu. With the Management Layer, you specify the parameters on the `Start Info` tab (in the `Command Line Arguments` field) of the LDS `Properties` window.

This section lists the required command-line parameters. See also the *Framework 7.2 Deployment Guide.*

Note: The first command-line parameter is always the name of the executable application file. On Windows, it is best to add the extension `.exe` to the executable file's name. For example, use `LDServer` on UNIX/Linux and use `LDServer.exe` on Windows.

These are the required command-line parameters:

`-host <confighost>` Represents the host running Configuration Server

`-port <configport>` Represents the port used by Configuration Server

`-app <appname>` Represents the name of the application as configured in Configuration Manager

`-l <license_file>` Represents either of:

The full path to, and the exact name of, the license file. For example, `-l /opt/mlink/license/license.dat`.

The `host name` and `port` of the license server, as specified in the `SERVER` line of the license file, in the format `port@host` . For example,
`-l 7260@ABCserver`.

Stopping LDS

You can stop LDS from the Management Layer, or you can use any of the following manual procedures:

- Use the `Ctrl + C` command in the component's console window (on both Windows and UNIX/Linux).
- Use the `End Task` button in the Windows Task Manager.
- Use the `kill <processnumber>` command on UNIX/Linux.

Chapter

5 Configuration Options

This chapter describes the configuration options and log messages specific to LDS. It contains the following sections:

Log options common to all servers are described in the "Common Log Options" chapter of any Framework T-Server document. Common log messages are described in *Genesys 7.2 Combined Log Events Help*.

LDS Section

You can find the LDS configuration options in a section called `LDS` on the `Options` tab of the `Properties` window of the LDS `Application` object.

license-file

Default Value: `license.dat`
Valid Value: Valid path to a valid license file
Changes Take Effect: Immediately if command-line parameter `-l` is not specified

Specifies the location of the license file from which LDS obtains the license, if the license file location is not specified on startup (using the `-l` command-line parameter).

stat-calc-threshold

Default Value: `1`
Valid Value: Any integer from `0-100`
Changes Take Effect: Immediately

Specifies the frequency with which LDS recalculates internal statistics to sort available Receivers according to their loading. Value `1` means LDS

recalculates with every transaction; value `10`, with every 10 transactions, and so on. The higher the value, the less frequent the recalculation.

active-context-limit

Default Value: `1000000`
Valid Value: Any integer from `1-1000000`
Changes Take Effect: Immediately

Specifies the default maximum number of active transactions that each Receiver can process simultaneously. You can also configure this option per Receiver in the `LDS` section in the Receiver application. If you do so, that value overrides the default value set in the LDS application properties.

cleanup-timer

Default Value: `60`
Valid Value: Any integer from `10-60`
Changes Take Effect: Immediately

Defines (in seconds) how often LDS sends query requests to T-Server to check the status of nonactive transactions.

context-remove-delay

Default Value: `5`
Valid Value: Any integer from `0-30`
Changes Take Effect: Immediately

Specifies the interval (in seconds) for which the `Connection ID` of a call is kept after T-Server reports `End-Of-Call`. This interval is used for call distribution in Inter Server Call Control (ISCC).

context-cleanup

Default Value: `30`
Valid Value: Any integer from `15-11520`
Changes Take Effect: Immediately

Defines the time (in minutes) that LDS waits after the last event is received for an active transaction before querying T-Server to check whether the call still exists.

Note: For pre-7.2.002 versions of LDS only, when you set the value of option `use-end-of-call` to `false`, the value assigned to the `context-cleanup` option represents a compromise between memory usage, CPU usage, and network traffic. Increasing the value of the latter option increases the amount of memory usage in LDS. Decreasing the value creates more network traffic and higher CPU usage in LDS and T-Server.

Genesys suggests that you set the value of this option to a small multiple (between `3` and `5`) of the average length of the calls in the system. But the actual figure you decide upon is affected by your

contact center's call profile, average length of calls, and other technical data.

When you set the value of `use-end-of-call` to `true`, this constraint does not apply.

count-active-context

Default Value: `false`
Valid Values: `true, false`
Changes Take Effect: Immediately

With value `true`, LDS counts the total number of active transactions for each Receiver and issues an alarm when the number of transactions defined in the `active-context-limit` option is reached.

ha-sync-level

Default Value: `0`
Valid Values:

`0`	The backup LDS does not receive events from T-Server until it becomes primary. Synchronization between primary and backup LDS is performed with each transaction. Genesys advises against using this value in production environments with high call volumes because messages can be lost in failover scenarios.
`1`	Both primary and backup LDS receive all events from T-Server. Synchronization between primary and backup LDS is performed with each transaction. Using this value increases network traffic.
`2`	Both the primary and backup LDS receive all events from T-Server. Synchronization between primary and backup LDS is performed with each T-Server event. Using this value increases network traffic as well as the resource usage of LDS.

Changes Take Effect: Immediately

Defines the level of synchronization between a primary and backup LDS application. Increasing the value of this option reduces the risk of event loss in a switchover/failover scenario, but increases network traffic.

Note: The `ha-sync-level` option applies to primary/backup LDS Applications in `hot standby` mode only, and (from release 6.5.3 onward) is automatically adapted to the LDS redundancy level. If you have configured LDS in `warm standby` mode, this option is ignored. The backup and primary LDS can start with different values set for this option (though Genesys does not recommend this), but the backup LDS automatically adopts the value configured in the primary LDS.

keep-taction-stat

Default Value: `5`
Valid Value: Any integer from `1-1440`
Changes Take Effect: Immediately

Defines the length of time (in minutes) that LDS maintains information about Receiver loading in order to perform even distribution.

msg-duplication

Default Value: `0`
Valid Value: Any integer from `0-10`
Changes Take Effect: Immediately

Defines the number of additional Receivers that receive duplicate event messages. With value `0` (zero), only one Receiver receives the event flow for each transaction. With value`1`, two Receivers receive the event flow for each transaction, and so on.

This option relates to multiple Receivers in `primary` mode and does not conflict with `HA Receiver` mode, in which the same event is sent to two Receivers.

Warning! Use extreme caution when changing the value of this option from its default setting of `0` (zero). Setting nonzero values increases availability. Please consult the documentation for your Receiver application to determine whether it supports nonzero values for this option.

no-context-distribution

Default Value: `first`
Valid Values: `none`, `first`, `all`

`none`	Any events that do not have a call context are not distributed to Receivers.
`first`	Events with no call context are distributed to the first Receiver to connect to LDS at startup. If this Receiver is disconnected, LDS selects another Receiver to receive such events.
`all`	Events with no call context are broadcast to all connected Receivers that are registered for the device in question.

Changes Take Effect: Immediately

Defines which LDS Receiver or Receivers (none, only the first Receiver, or all Receivers) should receive messages without a call context. Currently this includes all T-Server events without a `Connection ID`.

Note: With value `first`, the first Receiver assignment may be changed when the set of Receivers is changed; for example, when a Receiver is dynamically added or deleted.

update-timestamp

Default Value: `false`
Valid Values: `true`, `false`
Changes Take Effect: Immediately

Defines if LDS uses the current LDS time to update the `timestamp` attribute in all events sent to Receivers. With value `false`, all LDS clients receive the T-Server timestamp in events.

Note: LDS does not perform any kind of conversion between time zones.

use-end-of-call (removed in 7.2.002+)

Default Value: `true`
Valid Values: `true, false`
Changes Take Effect: At T-Server connect/reconnect

Defines whether LDS uses the End-Of-Call feature if T-Server supports it. When this value is changed dynamically, it takes effect either for new T-Server connections, or after existing connections are reconnected.

use-query-call

Default Value: `true`
Valid Values: `true, false`
Changes Take Effect: Immediately

Specifies whether, when working with Senders with no "call-end-support," LDS uses `QueryCall` to verify the existence of the call in the Sender (where supported).

With value `false`, no `QueryCall` request is made to the Sender.

link-by-originator

Default Value: `false`
Valid Values: `true`, `false`
Changes Take Effect: Immediately

With value `true`, LDS attempts to link a known in-progress interaction with a new one using `ThisDN`, `OtherDN`, or `ThirdPartyDN` information from the call events.

validate-sender

Default Value: `true`
Valid Values: `true`, `false`
Changes Take Effect: Immediately

With value `true`, only the client that requested connection to an available (connected) T-Server is accepted. With value `false`, the client for the disconnected T-Server is temporarily accepted, and LDS performs a standard connect procedure to the T-Server.

initial-receiver-read

Default Value: `true`
Valid Values: `true, false`
Changes Take Effect: Immediately

With value `true`, LDS reads the configuration of all Receivers at startup. LDS can then validate Receivers without additional requests for information from Configuration Server.

query-dn

Default Value: `lds-query-dn`
Valid Value: Any character string
Changes Take Effect: Immediately

Specifies the name of the DN that LDS uses to perform queries for calls.

query-timer

Default Value: `2`
Valid Value: Any integer from `2-60`
Changes Take Effect: Immediately

Defines the time (in minutes) between the receipt of the last event for an active transaction and an LDS query to T-Server to check whether the call still exists.

Note: For pre 7.2.002 versions of LDS, when you set the value of `use-end-of-call` to `true`, the value of `query-timer` supersedes the value of option `context-cleanup` because LDS uses a different procedure for querying calls in T-Server releases 6.5.3 and later.

Genesys recommends setting the value of `query-timer` to half (or less) that of the value set for the `context-cleanup` option.

background-processing

Default Value: `false`
Valid Values: `true, false`
Changes Take Effect: Immediately
With value `true`, LDS reads all connection sockets immediately, and places client requests within the input buffer, preventing LDS client disconnection through configured timeouts.

With value `false`, LDS processes all T-Server messages immediately, and delays the message queue associated with all LDS client requests, until there are no T-Server messages.

When LDS processes client requests from the message queue, requests are processed in the order they were received by LDS. In releases before 7.0.001, LDS processed multiple requests from one LDS client before proceeding to the request of another LDS client, and so on.

Note: Use of this option can negatively impact LDS processing speed.

background-timeout

Default Value: `60 msec`
Valid Value: Any integer
Changes Take Effect: Immediately

With Background Processing functionality enabled (option `background-processing` set to `true`), this option specifies the length of time (in milliseconds) that LDS pauses between processing successive sets of client requests in `Background` mode.

distribute-mode

Default Value: `auto`
Valid Values: `auto, load, proxy`

`auto`	With value `auto`, LDS selects its own mode depending on its actual configuration environment. With LDS configured in Configuration Manager with application type `T-Server` and one T-Server added on the LDS `Connection` tab, LDS operates in `proxy` mode. For all other possible configurations, LDS operates in `Load Distribution` mode.
`load`	With value `load`, LDS is forced into `LoadDistribution` mode.
`proxy`	With value `proxy`, LDS acts as proxy between T-Server and clients

Changes Take Effect: Immediately

Specifies the default distribution mode of LDS.

Note: Genesys recommends that you do not make changes without careful preparation, because the effect on client applications of changing between `Proxy` and `LoadDistribution` modes could cause unexpected effects.

register-guard

Default Value: `5`
Valid Values: Any integer from `0-30`
Changes Take Effect: Immediately

Defines the timeout (in seconds) between LDS issuing a `LinkConnected` event (or a consecutive `RegisterAddress` event) to the client and the beginning of distribution of transactions to this client.

register-mode

Default Value: `tproxy`
Valid Values: `tproxy, tserver`

Changes Take Effect: On LDS restart

Defines `TProxy` mode behavior in processing `RegisterMode`.

Note: Applicable for LDS in `TProxy` mode only.

strict-backup-name

Default Value: `false`
Valid Values: `true, false`
Changes Take Effect: Immediately

With value `true`, LDS does not accept clients requests for connection to non-running (or passive) T-Servers. With value `false`, LDS does accept such requests (pre-7.1 behavior).

intra-cluster-distribution

Default Value: `all`
Valid Values: `one, all`
Changes Take Effect: Immediately

Specifies whether to distribute transactions to one Receiver in a cluster (value `one`), or all Receivers (value `all`).

If a nominated Receiver in such a configuration fails, LDS distributes subsequent transactions to the next Receiver as per Warm Standby (that is, no synchronization).

ha-dly-switchover

Default Value: `false`
Valid Values: `true, false`
Changes Take Effect: Immediately

Determines whether the backup LDS delays switching into the `Running` state for Management Layer until it has been notified that target synchronization has completed. With value `false`, the backup LDS switches into `Running` state as soon as the backup link is established.

max-update-rate

Default Value: `100K`
Valid Values: See decription
Changes Take Effect: Immediately

Defines the settings that manage the HA link load during initial transaction synchronization between the primary and backup LDSs.

`0`	Primary LDS informs backup about the transaction target only when the transaction is either created or became active (that is, a transaction-related event is received).
Non-`0` value	In addition to updates for new and active transactions, the primary LDS will send the specified number of updates for

inactive nonsynchronized transactions to the backup LDS per second.

The option is defined using format *x*<K|M|G> and the following rules apply:

- A value *x* without a suffix instructs LDS to use that value in bps.
- A value *x* with suffix K instructs LDS to use that value in kbps.
- A value *x* with suffix M instructs LDS to use that value in mbps.
- A value *x* with suffix G instructs LDS to use that value in gbps.

The value defines overall traffic on the backup link. For transaction synchronization, LDS uses available "spare" bandwidth.

Example Primary/backup synchronization for 10 cps (calls per second) and 10 messages per call where ha-sync-level = 2 generates around 100 kbps of traffic (without overhead) on an HA link.

Note: The option is effective for a primary LDS running in HA Hot-Standby mode.

keep-ext-key

Default Value: No default value
Valid Values: Comma-separated list of extensions
Changes Take Effect: Immediately

Specifies a list of extensions that will be preserved when LDS removes ConnID-related extensions from events EventRegistered and EventAddressInfo during call cleanup routines.

LDS Options Configured in Receivers

Set the following two configuration options in the LDS section of the relevant Receiver application. The section name within the Receiver application must be LDS.

group-id

Default Value: 0
Valid Value: Any integer from 0-65535
Changes Take Effect: At LDS restart

Specifies whether this Receiver is part of a group. When you configure two or more Receivers with the same value for this option, LDS treats them as a single group and they receive the same event set.

Configure this option with caution. The fact that grouped Receivers receive identical call information has potential impacts on Receiver functionality; for example, duplicated records in Call Concentrator, calls being routed by two different URSs, and so on).

Note: You must ensure that both Receivers in the Configuration Manager primary/backup pair have the same value for the `group-id` option (if the option is configured). From release 7.1, the Configuration Manager primary/backup setting takes precedence.

loading-coefficient

Default Value: `100`
Valid Value: Any integer from `0-100`
Changes Take Effect: Immediately

Defines the relative loading coefficient for each specific Receiver for `Weighted Round Robin (WRR)` mode. For example, a set of three identical Receivers configured with loading coefficients `100`, `70`, and `30` receive 50 percent, 35 percent and 15 percent, respectively, of the total number of transactions.

If you use the default value (`100`) for all Receivers (`100`), you disable `WRR` and transactions are distributed in `Load Distribution` mode. If you set any nondefault value for any one Receiver (or Receiver group), you enable `WRR`.

Note: With value `0` (zero), a Receiver is excluded from transaction distribution. However, distribution to this Receiver is still possible, either when no other targets are available (for example, there are no more Receivers, or no more Receivers registered on a specific resource) or when this Receiver is available for noncontext distribution.

Log Messages

Table 2 lists the log messages specific to LDS.

Table 2: LDS Log Messages

Code	Type	Message
39500	STANDARD	The same Receiver [*text*] already connected to sender [*text*]
39501	STANDARD	Non T-Lib Receiver rejected
39502	STANDARD	Unknown Receiver [*text*] cannot be accepted
39503	STANDARD	Requested Sender [*text*] rejects connection
39504	STANDARD	Requested Sender [*text*] is not available
39505	STANDARD	Requested Sender [*text*] is not supported

Table 2: LDS Log Messages (Continued)

Code	Type	Message
39506	STANDARD	Requested Sender [*text*] unknown
39507	STANDARD	Unsupported Receiver type [*text*]-[*text*]
39508	STANDARD	Receiver [*text*]-[*text*] type conflict found, expected [*text*]
39510	TRACE	Sender [*text*] disabled
39511	STANDARD	Sender [*text*] removed
39512	STANDARD	Sender [*text*] connected
39513	STANDARD	Sender [*text*] disconnected
39520	TRACE	New transaction [*text*]
39521	TRACE	Transaction [*text*] sender changed
39522	TRACE	Transaction [*text*] disabled
39523	TRACE	Transaction [*text*] timeout expired
39524	STANDARD	Transaction [*text*] *2N+M* distribution detected
39530	TRACE	Unknown response from sender [*text*]
39531	STANDARD	Make Call request from Receiver [*text*] not supported
39532	STANDARD	Number of active transactions (number) on Receiver [*text*] is exceeded (number)
39533	STANDARD	Impossible to select the target
39550	STANDARD	Equal weighting profile ON (that is, WRR is not active)
39551	STANDARD	Varied weighting profile ON (that is, WRR is active)

Chapter

6 Common Log Options

Unless otherwise noted, the log configuration options that this chapter describes are common to all Genesys server applications and applicable to any Framework server component. This chapter includes the following sections:

Note: Some server applications also support log options that are unique to them. For descriptions of a particular application's unique log options, refer to the chapter/document about that application.

Mandatory Options

You do not have to configure any common log options in order to start Server applications.

Log Section

This section must be called `log`.

verbose

Default Value: `all`

Valid Values:

`all`	All log events (that is, log events of the Standard, Trace, Interaction, and Debug levels) are generated.
`debug`	The same as `all`.
`trace`	Log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels) are generated, but log events of the Debug level are not generated.
`interaction`	Log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels) are generated, but log events of the Trace and Debug levels are not generated.
`standard`	Log events of the Standard level are generated, but log events of the Interaction, Trace, and Debug levels are not generated.
`none`	No output is produced.

Changes Take Effect: Immediately

Determines whether a log output is created. If it is, specifies the minimum level of log events generated. The log events levels, starting with the highest priority level, are Standard, Interaction, Trace, and Debug. See also "Log Output Options" on page 48.

Note: For definitions of the Standard, Interaction, Trace, and Debug log levels, refer to the *Framework 7.2 Deployment Guide* or to *Framework 7.1 Solution Control Interface Help.*

buffering

Default Value: `true`

Valid Values:

`true`	Enables buffering.
`false`	Disables buffering.

Changes Take Effect: Immediately

Turns on/off operating system file buffering. The option is applicable only to the `stderr` and `stdout` output (see page 48). Setting this option to `true` increases the output performance.

Note: When buffering is enabled, there might be a delay before log messages appear at the console.

segment

Default Value: `false`
Valid Values:

`false`	No segmentation is allowed.
`<number> KB` or `<number>`	Sets the maximum segment size, in kilobytes. The minimum segment size is `100 KB`.
`<number> MB`	Sets the maximum segment size, in megabytes.
`<number> hr`	Sets the number of hours for the segment to stay open. The minimum number is `1` hour.

Changes Take Effect: Immediately

Specifies whether there is a segmentation limit for a log file. If there is, sets the mode of measurement, along with the maximum size. If the current log segment exceeds the size set by this option, the file is closed and a new one is created.

expire

Default Value: `false`
Valid Values:

`false`	No expiration; all generated segments are stored.
`<number> file` or `<number>`	Sets the maximum number of log files to store. Specify a number from `1`–`100`.
`<number> day`	Sets the maximum number of days before log files are deleted. Specify a number from `1`–`100`.

Changes Take Effect: Immediately

Determines whether log files expire. If they do, sets the measurement for determining when they expire, along with the maximum number of files (segments) or days before the files are removed.

Note: If an option's value is set incorrectly—out of the range of valid values— it will be automatically reset to `10`.

keep-startup-file

Default Value: `false`
Valid Values:

`false`	No startup segment of the log is kept.
`true`	A startup segment of the log is kept. The size of the segment equals the value of the `segment` option.
`<number> KB`	Sets the maximum size, in kilobytes, for a startup segment of the log.
`<number> MB`	Sets the maximum size, in megabytes, for a startup segment of the log.

Changes Take Effect: After restart

Specifies whether a startup segment of the log, containing the initial T-Server configuration, is to be kept. If it is, this option can be set to `true` or to a specific size. If set to `true`, the size of the initial segment will be equal to the size of the regular log segment defined by the `segment` option. The value of this option will be ignored if segmentation is turned off (that is, if the `segment` option set to `false`).

Note: This option applies only to T-Servers.

messagefile

Default Value: As specified by a particular application

Valid Values: `<string>.lms` (message file name)

Changes Take Effect: Immediately, if an application cannot find its `*.lms` file at startup

Specifies the file name for application-specific log events. The name must be valid for the operating system on which the application is running. The option value can also contain the absolute path to the application-specific `*.lms` file. Otherwise, an application looks for the file in its working directory.

Warning! An application that does not find its `*.lms` file at startup cannot generate application-specific log events and send them to Message Server.

message_format

Default Value: `short`

Valid Values:

`short`	An application uses compressed headers when writing log records in its log file.
`full`	An application uses complete headers when writing log records in its log file.

Changes Take Effect: Immediately

Specifies the format of log record headers that an application uses when writing logs in the log file. Using compressed log record headers improves application performance and reduces the log file's size.

With the value set to `short`:

- A header of the log file or the log file segment contains information about the application (such as the application name, application type, host type, and time zone), whereas single log records within the file or segment omit this information.
- A log message priority is abbreviated to `Std`, `Int`, `Trc`, or `Dbg`, for Standard, Interaction, Trace, or Debug messages, respectively.
- The message ID does not contain the prefix `GCTI` or the application type `ID`.

A log record in the full format looks like this:

```
2002-05-07T18:11:38.196 Standard localhost cfg_dbserver GCTI-00-05060
Application started
```

A log record in the short format looks like this:

```
2002-05-07T18:15:33.952 Std 05060 Application started
```

Note: Whether the full or short format is used, time is printed in the format specified by the `time_format` option.

time_convert

Default Value: `Local`

Valid Values:

`local`	The time of log record generation is expressed as a local time, based on the time zone and any seasonal adjustments. Time zone information of the application's host computer is used.
`utc`	The time of log record generation is expressed as Coordinated Universal Time (UTC).

Changes Take Effect: Immediately

Specifies the system in which an application calculates the log record time when generating a log file. The time is converted from the time in seconds since the Epoch (00:00:00 UTC, January 1, 1970).

time_format

Default Value: `time`

Valid Values:

`time`	The time string is formatted according to the `HH:MM:SS.sss` (hours, minutes, seconds, and milliseconds) format.
`locale`	The time string is formatted according to the system's locale.
`ISO8601`	The date in the time string is formatted according to the ISO 8601 format. Fractional seconds are given in milliseconds.

Changes Take Effect: Immediately

Specifies how to represent, in a log file, the time when an application generates log records.

A log record's time field in the ISO 8601 format looks like this:

```
2001-07-24T04:58:10.123
```

print-attributes

Default Value: `false`

Valid Values:

`true`	Attaches extended attributes, if any exist, to a log event sent to log output.
`false`	Does not attach extended attributes to a log event sent to log output.

Changes Take Effect: Immediately

Specifies whether the application attaches extended attributes, if any exist, to a log event that it sends to log output. Typically, log events of the Interaction log level and Audit-related log events contain extended attributes. Setting this option to `true` enables audit capabilities, but negatively affects performance. Genesys recommends enabling this option for Solution Control Server and Configuration Server when using audit tracking. For other applications, refer to *Genesys 7.2 Combined Log Events Help* to find out whether an application generates Interaction-level and Audit-related log events; if it does, enable the option only when testing new interaction scenarios.

check-point

Default Value: `1`
Valid Values: `0–24`
Changes Take Effect: Immediately

Specifies, in hours, how often the application generates a `check point` log event, to divide the log into sections of equal time. By default, the application generates this log event every hour. Setting the option to `0` prevents the generation of `check-point` events.

memory

Default Value: No default value
Valid Values: `<string>` (memory file name)
Changes Take Effect: Immediately

Specifies the name of the file to which the application regularly prints a snapshot of the memory output, if it is configured to do this (see "Log Output Options" on page 48). The new snapshot overwrites the previously written data. If the application terminates abnormally, this file will contain the latest log messages. Memory output is not recommended for processors with a CPU frequency lower than 600 MHz.

Note: If the file specified as the `memory` file is located on a network drive, an application does not create a snapshot file (with the extension `*.memory.log`).

memory-storage-size

Default Value: `2 MB`
Valid Values:

`<number> KB` or `<number>`	The size of the memory output, in kilobytes. The minimum value is `128 KB`.
`<number> MB`	The size of the memory output, in megabytes. The maximum value is `64 MB`.

Changes Take Effect: When memory output is created

Specifies the buffer size for log output to the memory, if configured. See also "Log Output Options" on page 48.

spool

Default Value: The application's working directory
Valid Values: `<path>` (the folder, with the full path to it)
Changes Take Effect: Immediately

Specifies the folder, including full path to it, in which an application creates temporary files related to network log output. If you change the option value while the application is running, the change does not affect the currently open network output.

compatible-output-priority

Default Value: `false`
Valid Values:

`true`	The log of the level specified by "Log Output Options" is sent to the specified output.
`false`	The log of the level specified by "Log Output Options" and higher levels is sent to the specified output.

Changes Take Effect: Immediately

Specifies whether the application uses 6.x output logic. For example, you configure the following options in the `log` section for a 6.x application and for a 7.x application:

```
[log]
verbose = all
debug = file1
standard = file2
```

The log file content of a 6.x application is as follows:

- `file1` contains Debug messages only.
- `file2` contains Standard messages only.

The log file content of a 7.x application is as follows:

- `file1` contains Debug, Trace, Interaction, and Standard messages.
- `file2` contains Standard messages only.

If you set `compatible-output-priority` to `true` in the 7.x application, its log file content will be the same as for the 6.x application.

Warning! Genesys does not recommend changing the default value of the `compatible-output-priority` option, unless you have specific reasons to use the 6.x log output logic—that is, to mimic the output priority as implemented in releases 6.x. Setting this option to `true` affects log consistency.

Log Output Options

To configure log outputs, set log level options (`all`, `standard`, `interaction`, `trace`, and/or `debug`) to the desired types of log output (`stdout`, `stderr`, `network`, `memory`, and/or `[filename]`, for log file output).

You can use:

- One log level option to specify different log outputs.
- One log output type for different log levels.
- Several log output types simultaneously, to log events of the same or different log levels.

You must separate the log output types by a comma when you are configuring more than one output for the same log level. See "Examples" on page 51.

Note: The log output options are activated according to the setting of the `verbose` configuration option.

Warning! If you direct log output to a file on the network drive, an application does not create a snapshot log file (with the extension `*.snapshot.log`) in case it terminates abnormally.
Directing log output to the console (by using the `stdout` or `stderr` settings) can affect application performance. Avoid using these log output settings in a production environment.

all

Default Value: No default value
Valid Values (log output types):

`stdout`	Log events are sent to the Standard output (`stdout`).
`stderr`	Log events are sent to the Standard error output (`stderr`).
`network`	Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the `all` log level option to the `network` output enables an application to send log events of the `Standard`, `Interaction`, and `Trace` levels to Message Server. `Debug`-level log events are neither sent to Message Server nor stored in the Log Database.
`memory`	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
`[filename]`	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends all log events. The log output types must be separated by a comma when more than one output is configured. For example:

```
all = stdout, logfile
```

Note: To ease the troubleshooting process, consider using unique names for log files that different applications generate.

standard

Default Value: No default value
Valid Values (log output types):

`stdout`	Log events are sent to the Standard output (`stdout`).
`stderr`	Log events are sent to the Standard error output (`stderr`).
`network`	Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database.
`memory`	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
`[filename]`	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the `Standard` level. The log output types must be separated by a comma when more than one output is configured. For example:

```
standard = stderr, network
```

interaction

Default Value: No default value
Valid Values (log output types):

`stdout`	Log events are sent to the Standard output (`stdout`).
`stderr`	Log events are sent to the Standard error output (`stderr`).
`network`	Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database.
`memory`	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
`[filename]`	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the `Interaction` level and higher (that is, log events of the `Standard` and

Interaction levels). The log outputs must be separated by a comma when more than one output is configured. For example:

```
interaction = stderr, network
```

trace

Default Value: No default value
Valid Values (log output types):

stdout	Log events are sent to the Standard output (stdout).
stderr	Log events are sent to the Standard error output (stderr).
network	Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database.
memory	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
[filename]	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels). The log outputs must be separated by a comma when more than one output is configured. For example:

```
trace = stderr, network
```

debug

Default Value: No default value
Valid Values (log output types):

stdout	Log events are sent to the Standard output (stdout).
stderr	Log events are sent to the Standard error output (stderr).
memory	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
[filename]	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the Debug level and higher (that is, log events of the Standard, Interaction, Trace, and Debug levels). The log output types must be separated by a comma when more than one output is configured. For example:

```
debug = stderr, /usr/local/genesys/logfile
```

Note: Debug-level log events are never sent to Message Server or stored in the Log Database.

Log File Extensions

You can use the following file extensions to identify log files that an application creates for various types of output:

- `*.log`—Assigned to log files when you configure output to a log file. For example, if you set `standard = confservlog` for Configuration Server, it prints log messages into a text file called `confservlog.<time_stamp>.log`.
- `*.qsp`—Assigned to temporary (spool) files when you configure output to the network but the network is temporarily unavailable. For example, if you set `standard = network` for Configuration Server, it prints log messages into a file called `confserv.<time_stamp>.qsp` during the time the network is not available.
- `*.snapshot.log`—Assigned to files that contain the output snapshot when you configure output to a log file. The file contains the last log messages that an application generates before it terminates abnormally. For example, if you set `standard = confservlog` for Configuration Server, it prints the last log message into a file called `confserv.<time_stamp>.snapshot.log` in case of failure.

  Note: Provide `*.snapshot.log` files to Genesys Technical Support when reporting a problem.

- `*.memory.log`—Assigned to log files that contain the memory output snapshot when you configure output to memory and redirect the most recent memory output to a file. For example, if you set `standard = memory` and `memory = confserv` for Configuration Server, it prints the latest memory output to a file called `confserv.<time_stamp>.memory.log`.

Examples

This section presents examples of a `log` section that you might configure for an application when that application is operating in production mode and in two lab modes, debugging and troubleshooting.

Production Mode Log Section

```
[log]
verbose = standard
standard = network, logfile
```

With this configuration, an application only generates the log events of the `Standard` level and sends them to Message Server, and to a file named `logfile`, which the application creates in its working directory. Genesys recommends that you use this or a similar configuration in a production environment.

Warning! Directing log output to the console (by using the `stdout` or `stderr` settings) can affect application performance. Avoid using these log output settings in a production environment.

Lab Mode Log Section

```
[log]
verbose = all
all = stdout, /usr/local/genesys/logfile
trace = network
```

With this configuration, an application generates log events of the `Standard`, `Interaction`, `Trace`, and `Debug` levels, and sends them to the standard output and to a file named `logfile`, which the application creates in the `/usr/local/genesys/` directory. In addition, the application sends log events of the `Standard`, `Interaction`, and `Trace` levels to Message Server. Use this configuration to test new interaction scenarios in a lab environment.

Failure-Troubleshooting Log Section

```
[log]
verbose = all
standard = network
all = memory
memory = logfile
memory-storage-size = 32 MB
```

With this configuration, an application generates log events of the `Standard` level and sends them to Message Server. It also generates log events of the `Standard`, `Interaction`, `Trace`, and `Debug` levels, and sends them to the memory output. The most current log is stored to a file named `logfile`, which the application creates in its working directory. Increased memory storage allows an application to save more of the log information generated before a failure. Use this configuration when trying to reproduce an application's failure. The memory log file will contain a snapshot of the application's log at the moment of failure; this should help you and Genesys Technical Support identify the reason for the failure.

Note: If you are running an application on UNIX, and you do not specify any files in which to store the memory output snapshot, a core file that the application produces before terminating contains the most current application log. Provide the application's core file to Genesys Technical Support when reporting a problem.

Log-Filter Section

This section must be called `log-filter`.

default-filter-type

Default Value: `copy`

Valid Values:

`copy`	The keys and values of the KVList pairs are copied to the log.
`hide`	The keys of the KVList pairs are copied to the log; the values are replaced with strings of asterisks.
`skip`	The KVList pairs are not copied to the log.

Changes Take Effect: Immediately

Specifies the default way of presenting KVList information (or, possibly, User Data) in the log. The selected option will be applied to all KVList pairs of the User Data, except the ones that are explicitly defined in the `log-filter-data` section.

Example

```
[log-filter]
default-filter-type=copy
```

Here is an example of a log using the default log filter settings:

```
message EventAttachedDataChanged
   AttributeConsultType     3
   AttributeOriginalConnID      008b012ece62c8be
   AttributeUpdateRevision      2752651
   AttributeUserData            [111] 00 27 01 00..
          'DNIS'                '8410'
          'PASSWORD'            '111111111'
          'RECORD_ID' '8313427'
   AttributeConnID              008b012ece62c922
```

Log-Filter-Data Section

This section must be called `log-filter-data`.

<key name>

Default Value: `copy`

Valid Values:

`copy`	The key and value of the given KVList pair are copied to the log.
`hide`	The key of the given KVList pair is copied to the log; the value is replaced with a string of asterisks.
`skip`	The KVList pair is not copied to the log.

Changes Take Effect: Immediately

Specifies the way of presenting the KVList pair defined by the key name in the log. Specification of this option supersedes the default way of KVList presentation as defined in the `log-filter` section for the given KVList pair.

Example

```
[log-filter-data]
PASSWORD=hide
```

Here is an example of the log with option `PASSWORD` set to `hide`:

```
message EventAttachedDataChanged
    AttributeConsultType         3
    AttributeOriginalConnID      008b012ece62c8be
    AttributeUpdateRevision      2752651
    AttributeUserData            [111] 00 27 01 00..
           'DNIS' '8410'
           'PASSWORD' '****'
           'RECORD_ID' '8313427'
    AttributeConnID       008b012ece62c922
```

Changes from Release 7.0 to 7.2

Table 3 shows the changes in common log configuration options between release 7.0 and the latest release 7.2.

Table 3: Common Log Option Changes from 7.0 to 7.2

Option Name	Option Values	Type of Change	Details
Log-Filter Section - New in 7.2			
keep-startup-file	false, true, <number> KB, <number> MB	New	See the description on page 43. **Note:** This option applies only to T-Servers.
default-filter-type	copy, hide, skip	New	See the description on page 53.

Table 3: Common Log Option Changes from 7.0 to 7.2 (Continued)

Option Name	Option Values	Type of Change	Details
Log-Filter-Data Section - New in 7.2			
<key name>	copy, hide, skip	New	See the description on page 53.

Chapter

7 High-Availability (HA) Configuration

This chapter describes high-availability (HA) modes for LDS and Receivers. It contains the following these sections:

LDS Backup Modes

Note: Genesys recommends that LDS pairs used in HA configurations have identical configurations.

LDS supports both warm-standby and hot-standby configurations.

Warm Standby

In LDS `warm standby` mode:

- Both the primary and backup LDSs are connected to all T-Servers using identical configurations.
- The primary LDS accepts Receiver connections and registers to T-Servers for events.
- The backup LDS does not accept Receiver connections and does not register for T-Server events.
- The `ha-sync-level` configuration option is ignored.

Hot Standby

In LDS `hot standby` mode:

- Both the primary and backup LDSs (ideally with identical configurations) are connected to all T-Servers.
- From LDS release 6.5.3 onward, the primary and backup LDS can start successfully with different values set in configuration option `ha-sync-level`. When both LDSs are started, the backup LDS adopts the value set for the primary LDS.
- Both the primary and backup LDSs accept Receiver connections and registration requests for T-Server events.
- The primary LDS registers to T-Servers to receive events according to Receiver requests. The backup LDS registers for events in the same way as the primary, but it also asks T-Server to mask events in a way that corresponds to the value set for configuration option `ha-sync-level`. See "Dynamic HA Model" on page 58.
- Only the primary LDS, passes events to Receivers and returns responses to T-Servers.
- Both the primary and backup LDSs synchronize transaction context in real time.
- To prevent network overloading, primary and backup LDSs are not synchronized at startup.
- The HA LDS Application Programming Interface (API) enables Receivers to connect to the primary and standby LDSs simultaneously. Registration requests go to both LDSs, and transparently to Receivers. Failover to the backup LDS also occurs transparently.

The standby mode for LDS can be different from the standby mode of T-Servers and of Receivers, and different T-Servers can have different standby modes.

Note: From release 6.5.3 onwards, where there is more than one LDS, each instance of LDS starts by default in `Backup` mode, and Management Layer must switch one instance to `Primary` mode.

Dynamic HA Model

Message-Synchronization Queue

From LDS release 6.5.3 onward, a Message-Synchronization Queue feature has been implemented to provide uninterrupted event flow from LDS to Receivers after LDS switchover. The queue enables the backup LDS to track

messages that the primary LDS has already sent. After a switchover, the new primary LDS resumes message distribution from the same point in the message queue at which the old primary LDS had stopped.

Changes to Redundancy Types

Table 4 shows how LDS reacts to dynamic changes to its redundancy type.

Table 4: Effect of Dynamic Changes to LDS Redundancy Types

	Redundancy Type Changed To...			
Redundancy Type Changed From...		Not Specified	Warm	Hot
	Not specified		Switch to backup `warm standby` mode. Wait for instructions from Management Layer.	Switch to backup `hot standby` mode. Wait for instructions from Management Layer.
	Warm	Backup LDS creates listen port. No action for former primary.		Start LDS synchronization. Backup LDS creates listen port.
	Hot	Stop LDS synchronization. Backup LDS resumes event distribution to clients.	Stop LDS synchronization. Backup LDS closes listen port.	

Changes to HA Synchronization Level

Table 5 shows the effects of dynamic changes to the value of configuration option `ha-sync-level`.

Note: These changes apply only to LDS in `hot standby` mode.

.

Table 5: Effects of Dynamic Changes to HA Synchronization Level

	Value Changed To...			
Value Changed From...		**0**	**1**	**2**
	0		*Primary:* No action.	*Primary:* Initialize message-synchronization queue.
			Backup: Set `Input Mask` to `High Level`.	*Backup:* Set `Input Mask` to `High Level`. Initialize message-synchronization queue.
	1	*Primary:* No action.		*Primary:* Initialize message-synchronization queue.
		Backup: Set `Input Mask` to `Low Level`.		*Backup:* Set `Input Mask` to `High Level`. Initialize message-synchronization queue.
	2	Primary: Clear message-synchronization queue.	*Primary:* Clear message-synchronization queue.	
		Backup: Set `Input Mask` to `Low Level`. Clear message-synchronization queue.	*Backup:* Clear message-synchronization queue.	

Receiver Backup Modes

LDS supports backup configurations for Receivers in `hot standby` mode and for CCon in `parallel` mode.

Chapter

8 LDS Support (Load Distribution Mode) of Routing

Note: The information in this chapter applies only to LDS in `Load Distribution` mode. See Chapter 1, "Overview," on page 13 for a description of this mode.

This chapter provides information on LDS support of Genesys Routing solutions (including Enterprise Routing and Network Routing). It contains the following sections:

The sections in this chapter explain:

- Routing (including high-availability routing) with LDS.
- The architecture supported by routing components, such as Universal Routing Server (URS) and Interaction Routing Designer (IRD).
- Application redundancy achieved by using LDS.
- Resource registration.
- Agent reservation features with LDS.

LDS and Routing

Use of LDS with Genesys Routing solutions provides a simple, scalable way to improve throughput by combining the processing power of multiple instances

of URS. The URS instances can run either on single-hardware platforms with multiple processors or on multiple-hardware platforms.

LDS Support of High-Availability Routing

The main reasons for using load distribution with Genesys Routing solutions in contact centers are:

- To meet customer requirements for several hundred interactions per second.
- To increase event throughput of a single URS process, which cannot be increased sufficiently by a hardware upgrade.
- To compensate for the decrease of single-URS throughput caused by complex routing-strategy requirements.
- To provide load distribution that is superior to multithreading, which is complex and does not solve all load distribution needs.
- To increase performance at a cost lower than that of a hardware upgrade.

For information on high-availability options for Enterprise Routing and Network Routing, see the *Universal Routing 7.2 Getting Started Guide.*

LDS and Routing Components

LDS and URS

URS performs the following tasks:

- Executes the rules specified within a strategy created in IRD.
- Creates a list of destinations or targets based on the strategy.
- Uses Stat Server statistics to determine the most appropriate target.
- Instructs T-Server where to route an interaction

URS connects to LDS for every T-Server in the LDS `Connections` list.

Each URS operates independently from other URSs. URS can operate when its `Connections` list contains connections to both LDS and T-Servers and it can have more than one LDS in its `Connections` list.

When disconnecting from LDS, URS attempts to connect to a backup LDS (if one has been configured), not to a backup T-Server.

LDS and IRD

IRD is a user interface for creating, editing, loading, and monitoring routing strategies. IRD communicates with URS through Configuration Server. Through a Message Server, IRD also receives from URS real-time routing information about interactions, server status, and Routing Points.

In a non-LDS environment, a strategy is loaded on a Routing Point in only one URS, even if there are multiple URSs monitoring the same Routing Point. However, to ensure consistent routing results in an LDS environment, every URS that is connected to the same LDS and is monitoring the same Routing Point must have the same strategy loaded for that Routing Point.

For information about using IRD, see *Interaction Routing Designer 7.2 Help.*

Using LDS in Routing Solutions

You can use LDS in Routing solutions to distribute requests among primary URSs and to combine the processing power of URSs to increase total throughput.

In configurations with multiple URSs, LDS distributes interactions among the URSs. This reduces the probability of a URS failure because of decreased processing on any single instance of URS. (For more information on load distribution, see "URS as a Client to LDS".)

When using LDS, you can achieve redundancy in a configuration with any number of single instances of URS with no dedicated backup URS. In this configuration, when one URS fails, LDS redirects new routing requests to the remaining URS(s). After Management Layer restarts the failed URS, LDS resumes load distribution to all servers.

Redundant configurations using LDS with multiple instances of URS offer a way to scale up Routing solutions when interaction volume exceeds the capabilities of the existing hardware platform.

For more information on redundancy using LDS, see "Scalability for LDS and URS Pairs" on page 66.

System Configuration and LDS

This section describes some possible ways of configuring LDS with Routing solutions.

URS as a Client to LDS

For load distribution in configurations with URS as a client to LDS, connections between T-Server, LDS, and URS occur in the following ways:

- One T-Server to one LDS to multiple URSs
- Multiple T-Servers to one LDS to ***n*** URSs, where ***n*** refers to the number of single instances of URS required to meet the specified load
- Multiple T-Servers to multiple LDSs to multiple URSs

One T-Server, One LDS, and Multiple URSs

Use a configuration with one T-Server to one LDS to multiple URSs when the transaction rate (interactions/second) on the switch/T-Server pair is higher than a single instance of URS can handle. LDS will distribute routing requests (balance the load) among URSs. You can put one or multiple instances of URS on the same computer with multiple processors, or you can distribute URSs on several computers when the contact center's call volume requires additional processing power.

Multiple T-Servers, One LDS, and Multiple URSs

You can also connect multiple T-Servers to one LDS and multiple URSs to meet the specified load.

Figure 6 illustrates a configuration with LDS at the center of message distribution from multiple T-Servers to multiple URSs.

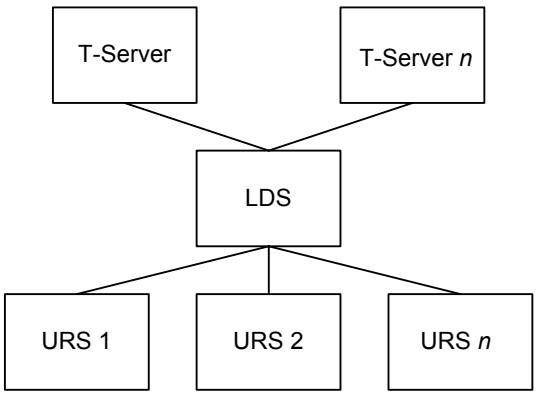

Figure 6: Multiple T-Servers to LDS to Multiple URSs

LDS distributes routing requests (balances the load) among the URSs. The multiple T-Servers can be either premise or network T-Servers.

You can also configure an environment using LDS and an ***n*** + 1 URS redundancy configuration without using a backup URS. (Here, ***n*** refers to the number of URSs required to meet the specified load.) When one URS shuts down, LDS redirects new routing requests to the remaining servers until Management Layer restarts the URS that shut down.

This form of redundancy (an ***n*** + 1 URS redundancy configuration) is most likely used in configurations with two or more primary URSs performing load balancing without LDS. However, LDS offers a growth path for an enterprise to add routing services when interaction volume increases beyond the capabilities of the existing hardware and software.

Note: Because detection and notification of URS failure are communicated through network messaging, you must configure network bandwidths to ensure that no delay occurs between SCS (Solution Control Server), URS, and LDS.

URS can receive and process messages from multiple T-Servers (including T-Server for PBX and Network T-Server) of any media type through the same LDS .

You do not have to install instances of LDS and URS on the same computer. However, you can strategically position LDS and multiple URSs on the same computer to optimize network traffic.

Multiple T-Servers, LDSs, and URSs

You can connect a URS to two different LDSs of the same type. This configuration might be necessary when the transaction rate of a single LDS exceeds its capacity; for example, if one LDS handles the transaction load for multiple T-Servers.

Figure 7 illustrates a configuration with multiple T-Servers dividing the transaction load between two LDSs and distributing transactions to multiple URSs. In this figure, URS 3 is also connected to two different LDSs of the same type.

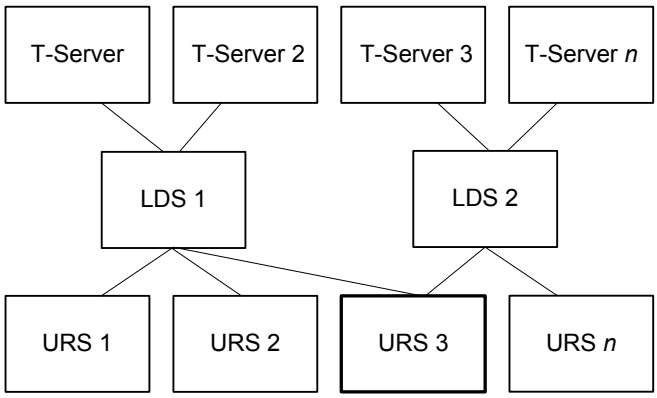

Figure 7: Multiple T-Servers to Multiple LDSs to Multiple URSs

LDS distributes routing requests according to the routing registration of URS. For information on this topic, see “Additional Information for LDS with URS” on page 68.

Note: Because LDS 1 and LDS 2 are not synchronized, there is no guarantee that the same router gets the call after a transfer. This may affect Inter Server Call Control (ISCC).

For information on a phased approach to introducing LDS into an existing architecture, see the *Universal Routing 7.2 Getting Started Guide.*

LDS and Network Routing

You can add a new URS and LDS to an existing infrastructure to handle extra call volume and to provide scalability when the network's interaction volume increases beyond the capability of the existing infrastructure.

Figure 8 illustrates the addition of a new URS and LDS to a Network Routing solution.

.

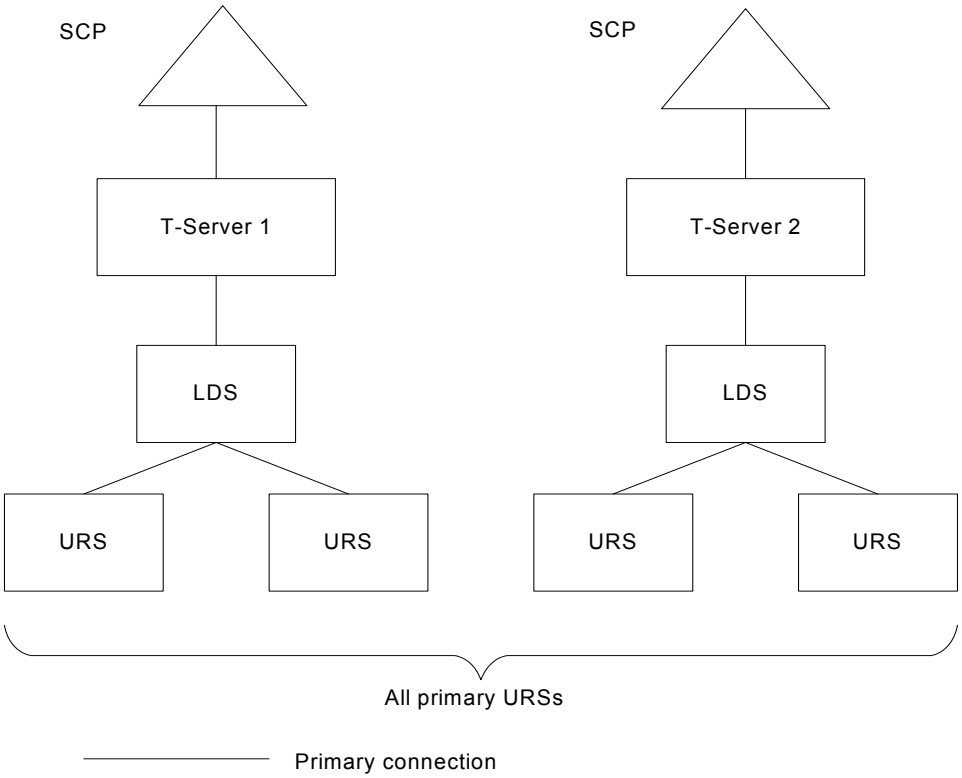

Figure 8: Adding a New URS and LDS to a Network Routing Solution

Scalability for LDS and URS Pairs

In Figure 9 on page 67, calls are distributed to 2*n* active servers, each handling approximately a 1/***n*** share of the load. Thus, the proportion of calls that cannot be routed when one server becomes unavailable is <= 1/(***n*–1**).

The connection between the `hot-standby` URS and its dependent servers is fully established. When a primary URS shuts down, new routing requests continue to be directed to the `hot-standby` URS in the pair. After the `hot-standby` URS becomes the primary URS, it replays the strategy for the pending interactions waiting in the Routing Point.

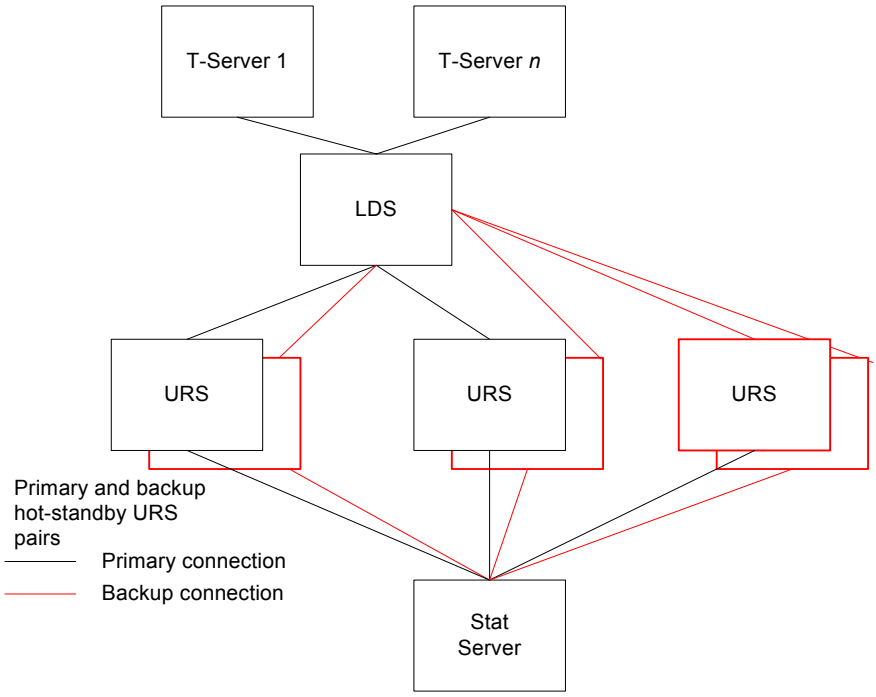

Figure 9: HA Routing Using an LDS and 2n URS Configuration

URS and Backup LDS

You can also configure LDS in primary and backup standby pairs. Only the primary LDS accepts connections from URS and registers for T-Server(s).

Backup LDS in Warm Standby

When a primary LDS shuts down, Management Layer notifies URS to reconnect to the backup LDS when the LDS failover process is complete.

Because a primary and backup LDS pair in `warm standby` mode are not synchronized, interactions in progress during failover are not recoverable. If there is a breakdown in communications between T-Server and URS during the failover period, the switch might default-route interactions.

Note: The fact that URS cannot route interactions *in this scenario* does not mean that they are irrecoverable. It means that URS cannot take control of these interactions or attempt to route them until after failover is complete.

For more information, see "High-Availability (HA) Configuration" on page 57.

Backup LDS in Hot Standby

You can also configure a primary and backup LDS pair in `hot standby` mode. Primary and `hot-standby` LDSs accept connections from URS and have connections to T-Server(s). However, only the primary LDS distributes events from T-Server(s).

Because primary and `hot-standby` LDSs are synchronized in a transaction context, T-Server events that the primary LDS submits are replayed by the `hot-standby` LDS after failover is complete. This means that no potential interactions are lost during the failover period.

See also "ha-sync-level" on page 31 and "Dynamic HA Model" on page 58.

Additional Information for LDS with URS

After you have configured and installed LDS, you can set up the routing environment. There are no routing-specific options that you need to set up in LDS or URS to enable routing to work with LDS: you have already set up options for LDS. (See page 29.) However, to get the most benefit from LDS, you need to take actions in respect of the following:

- LDS and Receiver type
- Resource registration
- Agent reservation options

LDS and Receiver Type

Currently, LDS is type specific. As a result, URS (the Receiver) can connect only to the LDS assigned to URS(s)—URS cannot connect to an LDS assigned to CCon.

LDS determines its type dynamically during runtimefrom the type of the first successful client connection to LDS; therefore, no explicit option settings are required.

Resource Registration

URS registers all Routing Points except those with option `event_arrive` configured and set to `none` in the Routing Point or virtual Routing Point properties. (This means that if you have not configured option `event_arrive`, URS registers for this Routing Point.) If URS is not registered to the specific Routing Point, URS receives no routing requests. Within a given LDS configuration, URS uses this mechanism to register to different Routing Points. Figure 10 illustrates this scenario.

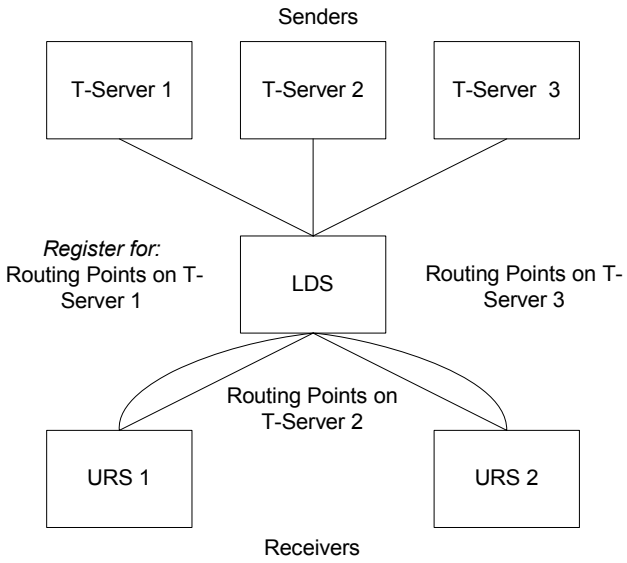

Figure 10: LDS and URS Registration of Routing Points

In Figure 10, URS 2 does not receive routing requests from T-Server 1. You establish this by configuring the `Annex` tab in the T-Server 1 `Application` object with a folder bearing the name of the URS 1 application at the individual Routing Point-level or virtual Routing Point-level.

LDS distributes the load (routing requests) according to the registration of Routing Points in URS. Interactions from T-Server 1 are sent only to URS 1. Interactions from T-Server 2 are distributed in `Load Distribution` mode between URS 1 and URS 2, and interactions from T-Server 3 are distributed only to URS 2. For load distribution of interactions from T-Server 2, the load on URS 1 and URS 2 (from processing interactions of T-Server 1 and T-Server 3) is taken into account.

Note: Using LDS does not affect Inter Server Call Control (external) routing functionality related to Routing.

Agent Reservation Options

Using LDS to achieve redundancy or load distribution always requires two or more URSs to route interactions from multiple T-Servers. In a configuration with LDS, the probability of having two URSs requesting the same routing target is higher. As a result, you must enable the Agent Reservation feature when using LDS.

Four options are related to the Agent Reservation feature at the URS application level (see the *Universal Routing Reference Manual* and the *Universal Routing Deployment Guide* for complete information about how these options function):

- `agent_reservation`
  This option is set at the URS Application level. It instructs URS to send a `reserve agent` request to T-Server and to wait for confirmation from T-Server before routing interactions to an agent.
- `transition_time`
  This option defines the minimum time (in seconds) that URS waits between the moment an interaction is routed to a target such as an agent, a place, or a DN, and any subsequent check for routing to the same target. To avoid repeated routing to the same target, you must set a nonzero value for this option.
- `reservation_pulling_time`
  This option temporarily eliminates the regular 2-second pause cycle for URS to select each routing target. With this option enabled, URS sends a reservation request for another ready agent immediately after a negative response to the preceding request. Enabling this option increases network traffic.
- `treatment_delay_time`
  This option delays treatment if `agent_reservation` is used.

Two options are related to the Agent Reservation feature at the T-Server application level. See the appropriate *Deployment Guide* for your specific T-Server for more details:

- `reservation_time`
  This option determines the time interval (in milliseconds) to reserve an agent. During that interval the agent cannot be reserved again. Use this option to handle race conditions caused by two or more routing requests for the same target. It does not work for multidirect (for example, direct transfers between agents) or multi-ACD interactions.
- `reject-subsequent-request`
  With value `true`, T-Server rejects subsequent requests for an agent reservation from the same client application for the same `Agent` object. With value `false`, a subsequent request prolongs the current reservation made by the same client application for the same agent.

Note: You must set the value of this option to `true` in all T-Servers that are LDS clients.

Chapter

9 LDS Support (Load Distribution Mode) of Call Concentrator

Note: The information in this chapter applies only to LDS in `Load Distribution` mode.

This chapter provides recommendations on using LDS and Call Concentrator (CCon). It contains:

Recommended Configuration

The recommended CCon configuration in an LDS (`Load Distribution` mode) environment comprises a redundant LDS process that receives T-events from all T-Servers, then distributes them to ***N*** + 1 CCon processes. (Here, ***N*** is the total expected traffic, divided by the traffic expected to be processed by a single CCon).

Additionally, you must set alarms in Management Layer against CCon's log events that indicate a database problem. It is also desirable to configure these alarms to automatically shut down the Call Concentrator process that reported the related event. Refer to Call Concentrator and Management Layer documentation for full details on how to do this.

This configuration ensures that if any Call Concentrator is closed down in this scenario, LDS automatically redistributes the load among the remaining processes that have enough processing power to handle it. Therefore, only the interactions in progress at the time of failure are affected.

Note: Setting the LDS configuration option `no-context-distribution` to value `all` allows Call Concentrator to populate `AREC` tables for non-call–related data, including custom states. Such data is most likely to be duplicated.

Index

D

E

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W